Starting from Scratch: Experimenting with Computer Science in Flemish Secondary Education

Francis wyffels
Electronics and Information Systems,
Ghent University
Francis.wyffels@UGent.be

Bern Martens
Secondary School Teacher Education,
Leuven University and Leuven University College
Bern.Martens@cs.kuleuven.be

Stefan Lemmens
Secondary School Teacher Education,
Leuven University College

ABSTRACT

In the Flemish secondary education curriculum, as in many countries and regions, computer science currently only gets an extremely limited coverage. Recently, in Flanders (and elsewhere), it has been proposed to change this, and try-outs are undertaken, both in and outside of schools. In this paper, we discuss some of those efforts, and in particular take a closer look at the preliminary results of one experiment involving different approaches to programming in grade 8. These experiments indicate that many students from secondary schools would welcome a more extensive treatment of computer science. Planning and implementing such a treatment, however, raises a number of issues, from which in this paper, we formulate a handful as calls for action for the computer science education research community.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education, curriculum*

General Terms

Human Factors, Experimentation

Keywords

Computer science education, flemish secondary education

1. INTRODUCTION

Over the last century, there has been a shift from an industry driven to an information driven society, processing ever increasing amounts of information. Therefore, people that are able to come up with processes to acquire, compute, store and transmit all this information in an automated way are in great demand. In Europe, the Information and Communication Technology (ICT) business currently generates 3.4% of the jobs and features 274,000 open positions, possibly icreasing to a million by 2020 [4].

Additionally, computers and other electronic devices are increasingly being part of every citizen's life. Despite this, only a limited amount of people seem to have even a basic understanding of the principles behind this technology. Moreover, as fun videos like "Teens react"[1] illustrate, this is not limited to "older" people. We believe that a lack of basic computer science (CS) knowledge results in limited use of the capabilities of computers, smart phones and other electronic (information processing) devices.

Another benefit of mastering computer science, is that with limited investments one can start a business. New devices such as smartphones are continuously in need for applications and with just a computer one can easily sell own coded applications worldwide. This already boosted the lives of many people, as initiatives such as Coders 4 Africa[2] illustrate.

Currently, Flemish K-12 computing education focuses almost exclusively on digital literacy [7]. Consequently, a vast majority of Flemish K-12 students gets very little exposure to computer science. However, some Flemish schools do experiment with CS education.

In this paper, we discuss some of those experiments briefly and one of them more extensively, and formulate some preliminary conclusions. Next, from these experiments, as well as from contexts such as the recent European Digital Skills framework [2], we formulate a number of issues in CS education that we believe to be in need of further research.

2. CS FOR FLEMISH STUDENTS

As discussed in [7], the exposure to CS in school for most Flemish students is currently limited to less than 20 hours of class in grades 9 and/or 10, and moreover suffers from many problems such as a lack of curricular ambition, sufficiently skilled teachers and suitable teaching materials. In this section, we present (Flemish) examples showing that one can do better than this.

2.1 Extracurricular CS activities

While CS education in Flemish schools is barely implemented, outside schools there are some opportunities for children and teenagers to get proficient. One example is

[1]Teens React to the 90s internet: what is a modem? (visited in June 2014)
[2]coders4africa.org (visited in June 2014)

WiPSCE '14, November 05 - 07 2014, Berlin, Germany
 ACM 978-1-4503-3250-7/14/11 ...$15.00
http://dx.doi.org/10.1145/2670757.2670763.

CoderDojo Belgium[3] which, in line with the international Coderdojo movement offers to children between 6 and 18 years old the opportunity to learn coding (and much more).

Another example is RoboCup Junior[4] which aims to promote STEM by means of an annual open robot competition for children between 8 and 18. Unlike CoderDojo, they do not offer any workshops for the children themselves. Instead, RoboCup Junior facilitates workshops for coaches. This approach is similar to that of Dwengo[5], a non-profit organisation that promotes CS (and STEM) through open software, hardware and educational packages. By means of local as well as international projects with teenagers and teachers, CS education is offered in an integrated context[6].

While these and other extracurricular activities present a partial remedy for the lack of CS education in schools, they fail to act as a full substitute for it. Firstly, most of the extracurricular activities are fully booked and cannot cope with the high demand. Consequently, most students are left out. Secondly, these initiatives rely mostly on the alertness of the parent (or teacher) to send the children. Therefore, various minorities and/or socially disadvantaged groups are clearly underrepresented. Thirdly, most of the initiatives only cover a part of CS (in most cases only coding). Consequently, since CS is much more than coding (see [1] for a review and [8] for an example implementation), the participants do not get to appreciate CS in its full breath.

These disadvantages of extracurricular initiatives can only be repaired through the incorporation of CS in the regular educational curriculum.

2.2 Exceptional schools

Despite the lack of CS in most official curricula, quite a few Flemish schools do implement more extensive CS education in various ways. Some schools are inspired by extracurricular approaches and organise e.g. a technical club during lunch breaks. Other schools use the curricular freedom Flemish education provides for two hours a week in grades 11 and 12 of general education to give an introduction to programming, do a robotics course, learn to make websites, etc.

There are also schools with mainly technically oriented study programmes which integrate substantially more programming and CS in their programme for grades 11 and 12 than the official curricula require. Interestingly, some schools experiment with older students (grades 11 and 12) tutoring younger students (grades 6 to 9) in (experimental) CS classes at their level.

So, quite a few schools use their creativity and curricular latitude to implement CS education to some extent. However, most of their efforts are limited to programming in grades 11 and 12.

Also, they rely heavily on the enthusiasm and expertise of individual teachers. In some cases where the former is much larger than the latter, this entails problems with the level of the content. Worse, the lack of any structural context for these efforts implies that they depend heavily on the availability of particular individuals in a school: if they leave, their course leaves with them.

Finally, in most initiatives known to us, participation by students is optional. As a consequence, they suffer from some of the same drawbacks as the extracurricular activities. This is illustrated by the fact that participation of female students is often low. Questioned on this point by one of the authors, the female students participating in the technical club mentioned above replied that they found the technical club interesting and enjoyed it a lot, but did not know why their female colleagues did not want to join.

Very recently, since September 2013, a few schools offer substantial CS courses in grade 7 and plan to structurally embed and extend this all the way up to grade 12. For the schools who started this already, this is part of a new study profile putting more emphasis on the T of Technology and the E of Engineering in STEM education (in general education). These initiatives have attracted quite a bit of attention in the Flemish secondary school "landscape" and inspired other schools to embark on similar endeavours. It is one of those schools that we discuss in the next section.

2.3 CS in the classroom: a coding experiment

From April to June 2014, we carried out an experiment in a Flemish school. Over a period of 6 weeks, 6 groups of students (131 in total of which 60% boys and 40% girls) in grade 8 (typical 13 years old) were taught 4 (consecutive) 100 minute classes on "An introduction to programming". The study profiles of the students involved were situated in general education focusing on mathematics, modern languages, classical languages and/or science.

After a joint unplugged introduction to the concepts of algorithms and programming, and a warming up exercise doing the problems on code.org[7], two groups continued with the graphical block based programming environment Scratch[8], two groups worked with Java using the (graphical) Greenfoot IDE[9], and the final two groups learned how to program a Dwengo robot using the Dwengo Blocks [10][10] programming language. To the best of our knowledge, it is the first time that an experiment like this was undertaken on this scale with students in this age group in Flemish education.

2.3.1 Some results from the teacher's point of view

The introductory class went well with all groups. The students enthusiastically "programmed" their teacher to make a jam sandwich (inspired by the British CAS example[11]), and quickly grasped some crucial lessons about the importance of sequence and precision in formulating an algorithm. Also the students liked the elementary coding exercises at code.org featuring e.g. Angry Birds.

The classes that continued with Scratch first had to implement a simple car racing game, and subsequently were invited to design and implement their own computer game, presenting it to the whole class group at the end of the fourth class. In general, this went well: most students were interested and motivated and many achieved nice results. One specific issue lies in the choice between a more directed and a more open approach. Scratch, with its low threshold, but ample advanced extension features, turned out to be very well suited for the latter approach. It did have the drawback however that the attention of the less motivated students

[3] coderdojobelgium.be (visited in June 2014)
[4] robocupjunior.be (visited in June 2014)
[5] dwengo.org (visited in June 2014)
[6] See for example the CErrobotics project in Argentina (visited in June 2014)
[7] code.org (visited in June 2014)
[8] scratch.mit.edu (visited in June 2014)
[9] greenfoot.org (visited in June 2014)
[10] blocks.dwengo.org (visited in June 2014)
[11] Program your teacher (visited in June 2014)

tended to wander away from Scratch and programming.

Since programming in a textual programming language is often perceived as something difficult, continuing with Greenfoot (in grade 8!) after the first class was considered as a rather risky experiment. We were, however, amazed at the success of this approach! Most students remained interested and motivated throughout and achieved impressive results (again coding a game of their own design after completing an introductory challenge offered by the teacher). Boys tended on average to be initially more excited (and relaxed) with the prospect of directly learning to code in a "real" programming language, but there was no significant difference in the average level of competence and quality of result achieved by boys and girls at the end of the four classes. Interestingly, one aspect did prove very difficult: in both Greenfoot groups the teacher tried to bring home the importance of code refactoring. This however proved to be beyond the intellectual capabilities of almost all students involved. Also, less motivated students tended to give up on the Greenfoot approach, more than with the other approaches.

The third approach was based on programming a robot to do stuff in the physical world, such as flashing lights in a particular pattern and following a prescribed course in the classroom. It turns out that learning to master a robot is perceived as highly motivating by the students (consistently more so than programming games with Scratch or Greenfoot). On the other hand, working with hardware moving about in the real world, introduces additional concerns for the (CS) teacher, such as the availability of sufficient copies of the hardware, and the effort to familiarize oneself with the electronics and robotics involved. With only two robots available for 20 students, both factors proved to be cumbersome in our experiment.

Interestingly, in none of the groups over the course of the entire experiment, any significant systematic differences in motivation or competence were noticed between boys and girls. However, in the game programming classes, most girls aimed for a game with a story (see e.g. also [6]), while most boys designed games with racing, shooting, etc. This was nicely illustrated by one girl who wanted to know whether in the introductory Scratch programming exercise, she was allowed to put a monkey on the race course rather than a car. We feel this aspect should be taken more fully into account in preparing teaching materials and textbooks.

2.3.2 *Some results from the students' point of view*

The participating students filled out a questionnaire at the start of the first and the third class, and at the end of the fourth class. About 16% of the students claimed to have some prior experience with programming. Surprisingly, 32% indicated having some experience with at least one in a given list of programming tools (Scratch 11%, Lego Mindstorms 14%). In other words, the prior use of some of these programming tools was not interpreted as "programming". Finally, programming was generally perceived as rather difficult: only 14% believed that they would be (or were) good at programming.

At the time the second questionnaire was filled out, differences between the three approaches became apparent (see Figure 1). While Scratch was perceived as fun and interesting, the textual programming approach of Greenfout was marked as difficult and challenging.

This was confirmed by the third questionnaire in which

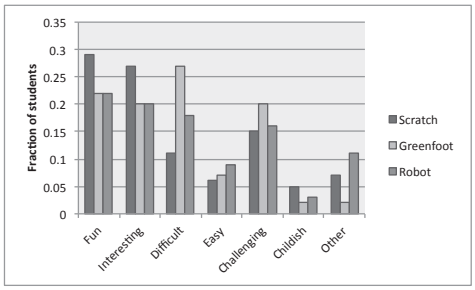

Figure 1: Results of questionnaire 2: students indicate their (per group) main feeling while programming.

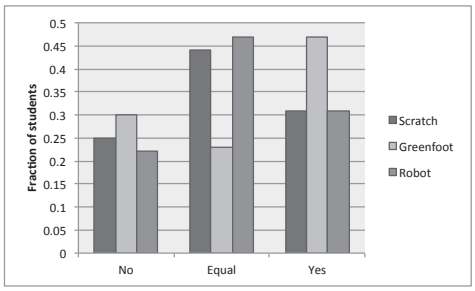

Figure 2: Results of questionnaire 3: students (per group) answer the question: "Is programming harder than expected?".

students had to indicate if they found programming harder than expected. As can be observed in Figure 2, almost half of the Greenfoot students thought so. In this group (not illustrated) 20% of the students found programming less fun than expected. However, also 17% of the Scratch group indicated that they experienced less fun than expected. Only 9% of the students working with the robot answered to have less fun, suggesting that working with a robot might be more generally appealing.

Finally, students were invited to indicate whether they would like to have a full programming course as a separate topic during the entire year. In all groups, such a course got the acclaim of about 40% of the students, and was rejected by about 30%. The school where the experiment was run has meanwhile decided to start with such a full programming course in grade 7. Moreover, they plan to develop computer science over the full six years of their curriculum. Our experiment was limited to programming, but one of the challenges in such an endeavour will of course be to broaden the scope to other aspects of computer science.

3. CALL FOR ACTION

Above, we discussed some Flemish initiatives and experiments with CS (programming) education. They show CS and CS education can get both teachers and students going, even while policy makers drag their feet. However, they also leave us with some questions. While we realize that Flanders is currently not among the world leaders in K-12 CS education, we feel that the full answers to these questions are not yet available, even in countries which are. In this section, we therefore formulate four calls for (research) action, address-

ing issues that we believe to be important for the further development of CS as part of the core K-12 curriculum.

3.1 Classifying and measuring computer science skills

The digital competence of the European population is classified by the EU into five different competence areas: information, communication, content creation, safety, and problem solving [2]. While the area "problem solving" suggests CS, the available indicators (finding a job online, internet banking, connecting and installing new devices,...) show that it only covers digital literacy. In fact, only one indicator, "writing a computer program using a specialised programming language", targets one aspect of CS. If we want to improve the treatment of CS in education, and clarify how it relates to digital literacy, we need sufficiently precise metrics for identifying core CS skills and competences.

3.2 How to teach computer science

Perhaps more than in many other topics, students of all ages bring very diverse levels of competence in CS to the classroom. Currently, in Flemish secondary schools, one can find students in grade 7 with multiple years of programming experience while others (in fact most) have no CS related competences at all. While there is already some research on how to approach heterogeneous groups, we believe that more extensive research is necessary, especially with respect to CS education.

Over the last few years, multiple didactic CS tools have been developed. This includes hardware such as robot platforms [9], coding environments, and various educational games that may improve the learning experience and effectiveness in CS classes. Consequently, we now have a wide choice of tools and approaches, but no "best ones" have currently emerged. More research such as [5] should therefore be performed. Our own experiment described in Section 2.3 also provides a contribution.

3.3 When to start computer science education

In [3] six arguments for teaching science to young children are given. But what about CS? In the information age, information processing is all around us and part of nearly everyone's life at an early age. Does this imply that computer science education should also start as early as possible? And if so, how can this be further developed and implemented?

3.4 Evaluating the long-term effects of computer science education

Often it is claimed that more and better CS education will improve problem solving skills, lead to less unfilled positions in ICT, result in more and more diverse ICT professionals (e.g. more females and a better cultural mix), and help people to use computers and other information processing devices better and more safely. However, such claims remain vacuous without (more) research to corroborate them.

4. CONCLUSIONS

Computer science is currently largely lacking from the official Flemish education curricula. Nevertheless, inspired by extracurricular projects and increased public demand, some schools have started to implement CS education. In this paper, we discussed some of these experiments and derived some calls for research that would support the further development of K-12 CS education.

5. ACKNOWLEDGMENTS

The authors thank Ben Alen and Bart Demoen for their advice, feedback and support.

6. REFERENCES

[1] P. J. Denning. Great principles of computing. *Communications of the ACM*, 46(11):15–20, 2003.

[2] DG-CONNECT-F4. Measuring digital skills across the EU: EU wide indicators of digital competence. Technical report, European Commission, 2014. `http://ec.europa.eu/digital-agenda/en/news/measuring-digital-skills-across-eu-eu-wide-indicators-digital-competence`.

[3] H. Eshach and M. N. Fried. Should science be taught in early childhood? *Journal of Science Education and Technology*, 14(3):315–336, 2005.

[4] K. Gareis, T. Hüsing, S. Birov, I. Bludova, C. Schulz, and W. Korte. E-skills for jobs in Europe: Measuring progress and moving ahead. Technical report, European Commission, 2014. `http://ec.europa.eu/DocsRoom/documents/4398/attachments/1/translations/en/renditions/pdf`.

[5] B. Gibson and T. Bell. Evaluation of games for teaching computer science. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*. ACM, 2013.

[6] C. Kelleher, R. Pausch, and S. Kiesler. Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1455–1464. ACM, 2007.

[7] B. Martens and T. Hofkens. Positioning computer science in Flemish K-12 education: a reflection. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*. ACM, 2013.

[8] A. Tucker, D. McCowan, F. Deek, C. Stephenson, J. Jones, and A. Verno. A model curriculum for K-12 computer science: Report of the ACM K-12 task force computer science curriculum committee. Technical report (revised edition), ACM, 2011. `http://www.csta.acm.org/Curriculum/sub/CurrFiles/K-12ModelCurr2ndEd.pdf`.

[9] C. Vandevelde, J. Saldien, C. Ciocci, and B. Vanderborght. Overview of technologies for building robots in the classroom. In *International Conference on Robotics in Education, Proceedings*, pages 122–130. Lodz University of Technology, 2013.

[10] F. wyffels, K. Bruneel, P. Bertels, M. D'Haene, W. Heirman, and T. Waegeman. A human-friendly way of programming robots. In *5th International Workshop on Human-Friendly Robotics, Abstracts*. IEEE, 2012.